CHAPTER

7

Manage

Agile Operations: Powering the Modern Software Factory

In traditional enterprise IT, developers code and operations manage what comes "over the wall" to production. While DevOps regards this as the ultimate divisive anti-pattern, this practice has still been conducted for decades—but why?

The model generally persists because of the nature of customer engagement. Applications have been generally designed "inside-out," with customer interaction through a single channel. Even if the channel is digitized, the focus is on improving business efficiencies, with customer benefits only considered as an afterthought. For IT operations, supporting this model has been difficult but manageable.

But all this is radically changing. Now businesses understand that customer needs "outside" their organization must be brought "inside" and supported via omnichannel engagement. Omnichannel is all about continuity of experience, regardless of where, when, and how a customer interacts with a business.

From a commercial perspective, omnichannel provides an opportunity to enhance the all-important customer experience via new digital touch points; however, this increases IT operations complexity, with teams now faced with managing increased volumes of rapidly changing software services, delivered over modern and legacy applications and infrastructure.

A. Ravichandran et al., *DevOps for Digital Leaders*, DOI 10.1007/978-1-4842-1842-6_7

With companies digitally "upshifting" from business efficiency to business model transformation, the value proposition of IT operations must change—from being good at managing the technology status quo, to becoming more *agile* and integral to driving successful business outcomes.

This notion of an agile operation is highly synergistic with DevOps since it involves teams working collaboratively to establish a high-quality customer experience across the software lifecycle. Rather than wait until production and retrofit performance, an agile operations team works closely with development using new monitoring approaches to "bake" or craft quality into applications—as they're engineered, tested, and released.

Before examining new challenges and agile operations monitoring strategies, let's examine a case study where DevOps style practices have been used to great effect.

Case Study: ANZ Bank

Today, the only source of competition in the banking world is an obsession with serving and delighting customers. This is something Melbourne, Australia-based ANZ Bank had in mind recently when it launched a new application performance management (APM) program with the hopes of extending it across the entire IT infrastructure, which includes four data centers, mainframes, and more than 10,000 servers.[1]

"The primary goal of all my teams is 100 percent availability of services for our customers. It's simple as that," explains Adam Cartwright, head of IT Security and Operations at ANZ Bank.

"That means *whatever channel it happens to be*—whether it's a corporation doing a payroll transaction or a private user transferring money using Internet banking—it's got to be up. It's got to be seamless. It's got to perform to their expectations. That is the primary mission of operations within technology."

Unfortunately, the complexity of modern distributed applications means that this doesn't happen all the time, says Cartwright. Applications and systems go down, adversely impacting end users. In 2012, it became apparent that infrastructure monitoring alone—which focused on platform and event monitoring at the infrastructure level—was not enough to give ANZ Bank the insight it needed to fix and prevent incidents within the organization.

"What you really need to do is to understand the transaction flow within the application context you're going to identify the root cause, or if you're going to get early signaling of potential problems before a customer actually has [an incident]," he explains.

---

[1] Full story: http://www.ca.com/us/rewrite/articles/management-cloud/customer-obsessed.html

For example, the bank has a distributed payment application that it used to transact billions of dollars for its highest value customers. The application comprises more than 120 distributed servers and 60 separate Java applications. Whenever there was an issue with that application, it was nearly impossible to determine the root cause of the stability issues. The worst part was that many times ANZ Bank IT didn't realize there was a problem until its customers notified them about the issue.

These challenges led to many of the goals of ANZ Bank's Application Performance Management (APM). First and foremost, IT wanted to reduce the number of incidents caused by application releases. This would increase the quality and confidence of application deployment or changes while improving the overall lifecycle of new ANZ applications.

In addition, Cartwright wanted a way to perform deep dives into the application layers to obtain code-level visibility to measure performance and availability. He also wanted to measure and analyze transactions as they moved across the distributed and highly diverse ANZ Bank infrastructure technologies. These efforts have helped the bank minimize customer downtime.

Using APM, IT is now proactively alerted when there's an issue before it affects service. Since APM allows the company not only to look at transactions flowing through an application, but also to identify the business user behavior attached to those transactions, outages can be found more quickly, and applications can be redesigned so outages don't repeat themselves.

"Getting that insight in production is fantastic, but what is perhaps more brilliant is getting that *insight in the development-and-test environment*," he says. "We have been able to show that by putting APM into development and test—making it part of the process in those areas—you can stop defects from getting into production. And that's probably the most surprising thing to people new to APM, but the most critical thing from a production support point of view."

Indeed, says Cartwright, this is one of the more astounding benefits of APM. For example, one project team was able to prevent 10 high-severity incidents from happening, saving more than a dozen hours of investigations. Another project team reduced recovery time from more than four hours to less than 30 minutes with no impact to service. "While it's good to have APM and to use it to diagnose problems once they've occurred in production, you've got a customer that has been affected," Cartwright explains. "It's much better to stop that customer effect from occurring in the first place by not letting that sort of design issue propagate into production under transition."

One of ANZ Bank's first projects in the test–and-development realm was a payments application that caused serious production issues. The bank had five or six major releases each year, and every time a new version went live there was "a raft" of high-severity incidents. "No matter what we did with testing, reviewing, traditional sorts of approaches, or performance and volume testing,

we always managed to end up in a situation where we had a release go live and we'd have problems in production," says Cartwright.

Using APM, ANZ Bank eliminated between 10 and 15 high-severity incidents, which often stretched out three or four weeks following a release. "Now, we're down to one or two or, in some cases, zero [incidents]," says Cartwright. IT employees are thrilled with the change.

"Since APM has been in, we have been able to pinpoint the exact impacted servers and restart these without impacting business and payments processing," explains Joseph Rocco, Support Transition Analyst at ANZ. "Before APM, the LMS [Limits Management System] recovery was four-plus hours. Post APM, it has been around 30 minutes total to restart impacted servers with no outages."

This is an example of the fact that, while ANZ's main focus was on customer satisfaction, a positive side effect of reducing incidents is the boost it gave the IT organization as a whole. "If you can stop having incidents, you track capacity and headcount within the organization to do other things, more proactive things," Rocco says. "APM essentially means your organization is going to continue to grow as your system footprint grows and the complexity grows." IT employees are freed up to be proactive rather than reactive.

Following on from this success, ANZ extended its *shift-left* approach to more than 20 applications. As a result, it has code-level visibility of performance and availability issues, which not only stops defects from getting into production but also increases confidence in application development and deployment at the bank.

Both development and operations teams undertake performance and load testing and correlate their data. As a result, pre-production efficiency has increased with a 60 percent reduction in time spent on solving software problems, which equates to savings of AU $300,000.[2]

More Change, More Complexity

Progressive organizations like the ANZ Bank understand how customers expect rapid software iterations of new functionality together with high levels of performance. This fact was illustrated in an Enterprise Management Associates (EMA) report, which indicated that two-thirds of organizations who have embraced continuous delivery release code weekly or even more frequently.[3] But this is not without its problems, with the EMA report also suggesting that development now spends as much time supporting production as it spends writing new code, while operations spends more time on application support than on any other single task.

---

[2] https://www.brighttalk.com/webcast/7819/134027

[3] "Omnichannel, Microservices, and Modern Apps," January 2016: http://www.ca.com/content/dam/ca/us/files/white-paper/ema-omnichannel-microservices-and-modern-applications.pdf

This support now extends to managing modern microservice style architectures. Designed to be deployed as discrete elements (or services) performing a specific set of tasks and running as its own process, microservices break down specific functions into small components connected via APIs.

While this approach potentially allows services to be updated more regularly without impacting other elements supporting a business process, there are major operational challenges. Not the least:

- *Increased diversity*—With microservices, developers can code in multiple languages and work with databases best suited for their service. For operations, this means maintaining performance and availability over unfamiliar technologies like Node.js and MongoDB.

Tip Always remember that with modern digital systems applications supported by microservices and newer technologies, application performance management solutions must be more resilient and scalable than whatever they are monitoring!

- *Massive complexity*—One monolithic application can become thousands of microservices. Unlike monoliths, microservices make visualizing application topologies and transactional flow using traditional tools extremely challenging. Add the prospect of potentially running multiple service versions in parallel and monitoring complexity increases exponentially.

  Much of the microservice complexity lies in the relationship and API-centric communication between services. With distributed systems like these, teams must consider a whole range of new issues, including network latency, asynchronous messaging, and load balancing, not to mention end-to-end performance issues when microservices connect with back-end applications.

- *Increased noise*—With mobile apps, microservices, and containerized environments, the volume of alarms and alerts can grow significantly. Trying to find filter out noise and find the root cause of problems using traditional rules-based approaches becomes much more difficult.

■ **Tip** Consider periodically assigning developers to review the alerts and log messages their code is producing. This could be a useful way of identifying where refactoring work or improving application supportability is required.

- *Ephemeral nature*—With the pace of change necessitating far shorter application lifespans, triage teams no longer have the luxury of capturing and analyzing historical data using a plethora of tools. Teams require methods to better understand real-time performance across modern architectures (including containers), which in more dynamic environments might be changing in a matter of minutes, even seconds.

New IT Operations Imperatives

IT operations as a discipline will no longer be judged on how effective it is at fixing application problems, but rather on the ability to improve business outcomes—detecting and fixing issues, yes, but working collaboratively with other teams across the software factory to establish quality.

What's admirable in the case of ANZ is how managing to business outcomes has become an established part of the IT operations mantra. True, the team is still responsible for maintaining stability and resilience, but by establishing performance monitoring in areas beyond their traditional control, quality and confidence have increased substantially.

Rather than reactive break-fix approaches to monitoring, a DevOps focused operations function will leverage advanced tools to proactively ensure a quality customer experience before a system reaches production. In this sense, practitioners will become uber sysadmin *craftsmen* and as agile as their development colleagues. And with developers increasingly empowered to make operational decisions, a move toward the *agile operations* approach will become more important.

So what new skills will agile operations teams need to acquire? There are a few important ones to consider, covered in the following sections.

Proactive Engagement

The IT operations landscape changing is dramatically with many advances in technology, but not necessarily changes in values or thinking. With the democratization of operational functions, a good agile operations focused

practitioner will be one that strives to intimately understand the behavior of applications, nurture production systems, and feedback information and knowledge. To this end, agile operations craftsmanship will be less about pulling out router cables and console watching and more about analyzing app behavior and the customer experience to drive improvements.

Designing for Failure

The traditional approach of measuring operational effectiveness in terms of preventing failure doesn't work anymore. With cloud applications, there are many moving parts, in terms of technology and process. There will be mobile apps and APIs supporting new digital channels, but also back-end data and systems. In these environments, failures are inevitable, so the objective should be to design for them—containing problems, but still keeping the business running.

Accepting this reality, agile operations will work closely with development to mature the software engineering and monitoring practices needed to optimize modern cloud-based systems—such as, for example, establishing infrastructure monitoring with every release or exposing performance diagnostics with every application build.

Tip Make it a rule rather than the exception to establish infrastructure and performance monitoring in every environment—right from the moment something is provisioned or placed in maintenance mode.

Moving Beyond Resilience

Like the mythical Phoenix, modern cloud systems and microservices should be designed to bounce back from every situation. And they must, because when a business uses these approaches to engage customers at scale and deal with unknown demand, there'll be much more complexity—at the very least ensuring hundreds (perhaps thousands) of services continue to be available and performant.

Throw in complex architectural issues like asynchronous messaging and API latency and there's a whole new world of pain. It's analogous to cutting a head off the mythical Hydra. Just when you've addressed one problem, two more grow in its place. This is why resilience doesn't go far enough.

When considering new digital systems-of-engagement, applications can't only be resilient, they too have to be Hydra-like. Rather than fixing problems, agile operations methods work to improve applications both technically

and commercially after every significant event (positive or negative). This is especially important for mobile apps and why experience-based analytics is becoming increasingly important.

Making Support a Top Design Issue

Rather than working as downstream production traffic cops, agile operations will engage upstream with developers in an advisory capacity. Of course this requires development to be fully involved; having the desire to learn from IT operations in order to design and build systems that are much easier to support once in production.

But taking the time to learn is challenging if operations provides no useful information or the tools used only address monitoring from one perspective. It'll be difficult too, if in the rush to meet project level goals, development teams select their own point tools and methods at the expense of overall system-level resilience and performance.

Combatting these issues requires ending the "toe-to-toe" battles with development at various checkpoints across the software lifecycle (which never end well). It involves demonstrating how the knowledge and expertise everyone has acquired over many years not only helps improve application supportability, but also makes peoples jobs easier and more rewarding. Some examples include:

- In a mobile app scenario, presenting live usage crash analytics that developers can use to identify where functional improvements may be needed.
- Sharing APM toolsets that enable developers to review issues from their perspective. Serving up information in their terms, about their world, with their code.

Tip Consider organizing joint dev and ops workshops where teams openly discuss and share the "cool" things they've learned over many years. This includes operations sharing information about improving resilience and development explaining the methods used to release software updates in small batches.

- Openly discussing how older style alert and static baselining make less sense in monitoring modern dynamic environments and only increases the support burden.

- Jointly conduct triage scenarios, simulating what's really involved when attempting to fix application issues at 3:00am (clinically review tools to determine whether any of the "noise" actually warrants getting support staff out of bed in the small hours)!

Tip Use workshops and tools to: 1) help developers understand the "on-call" support implications of their designs, and 2) help operations understand what information developers need to make performance improvements.

Active Monitoring

Teams need to embrace active monitoring methods to build an understanding about issues before they affect customers. Part of this involves finding better ways to remove misleading alarms and false-positives.

Traditionally, monitoring solutions have dealt with false-positive alerts using performance baselines. Although this has helped, these approaches typically look at only one part of the issue: severity. A different way to look at the issue is to analyze both *severity and duration*. For example, a minor issue occurring over a long period of time could eventually escalate into a larger issue that teams need to investigate. Alternatively, a medium issue that occurs some of the time should raise an alarm because it could become a larger problem very quickly.

Tip Once techniques to remove irrelevant noise and alerts have been applied, look for ways to combine active alerting with methods to capture more detailed information. For example, automatically initiating a diagnostic transaction trace.

Toward Agile Operations

The 2015 Freeform Dynamics report, produced in association with CA Technologies, indicated that with DevOps, 63 percent and 61 percent of advanced adopters, respectively, were better able to help the business act swiftly on digital opportunities and attack and defend more quickly—with 77 percent and 72 percent also indicating improvements in achieving customer acquisition and retention goals.[4]

---

[4] https://www.ca.com/us/rewrite/articles/devops/assembling-the-devops-jigsaw.register.html

This suggests organizations can move fast without sacrificing high quality.

Achieving this requires IT operations adopting many practices and processes familiar to their agile development colleagues. Of course, this still requires monitoring the performance of production applications, but it also means ensuring information gained is fed back and incorporated into agile development processes (e.g., agile sprints). See Figure 7-1. But, and as we've described, IT operations must ensure that the information being shared with development is of higher value than that provided through traditional system alerting.

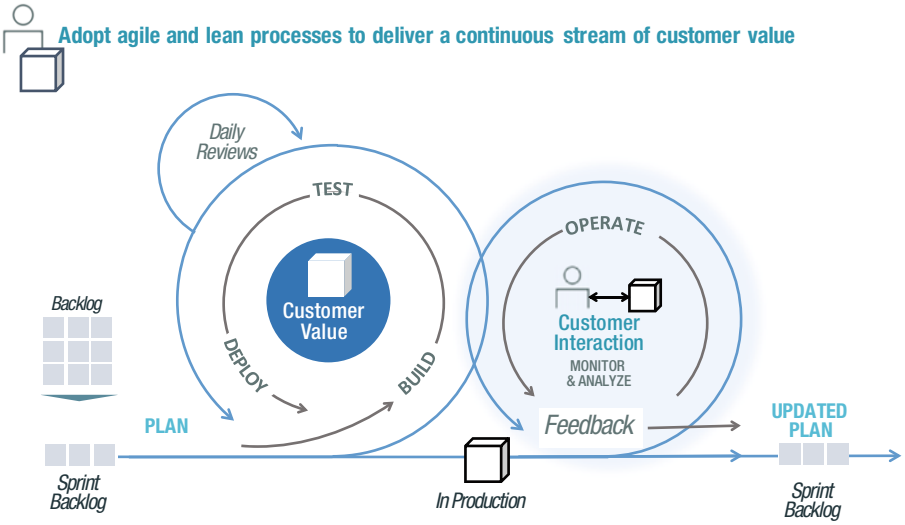

Figure 7-1. Agile development and operations

From a development perspective there are many new tools and techniques that support shifting-left phases so teams can work in parallel to meet the goals of continuous delivery. For example, and as described in Chapter 5, by simulating dependent systems, service virtualization solutions help teams remove constraints, thereby providing teams immediate and realistic test environments.

IT operations has to a lesser extent been slower adopting this approach. This is often due to keeping watch over a myriad of diagnostic tools—meaning less time is spent delivering information that's actually useful to their colleagues.

But failing to embrace an agile operations shift-left approaches can lead to cost increases and missed business opportunities. For example, hurriedly purchasing additional capacity due to unexpected performance problems, or hiring more contractors because of increased staff burnout and turnover after excessive after hours support.

Shift-Left Monitoring

Adopting agile operations methods like shift-left can be a daunting prospect when people feel ill-equipped to cope with any type of change. According to an Enterprise Management Associates (EMA) report, fewer than 50 percent of IT professionals are confident that their application management solutions can adequately meet the monitoring requirements of modern IT environments.[5]

It's important therefore to resist the urge to add another tool (be that commercial or open source) to a growing arsenal of monitoring tools. This approach can further increase team fragmentation, as confirmed in an Infrastructure & Operation Trends Survey, where 80 percent of respondents agreed that disjointed, cross-platform management leads to lost opportunities.[6]

As the survey alludes, this can be suboptimal from a business perspective, especially when incomplete (or incomprehensible) performance signals result in bad decisions. For example, snap purchasing servers due to an unexpected performance condition. This might have addressed the immediate issue, but the organization has just reduced its profit margins and the real problem still lurks-somewhere.

This suggests that monitoring approaches should be conducted from an "avoidance" perspective. That is, avoid problems by detecting them early and often, before they can impact the business. Do this with fewer tools or specialists and cost encumbrances are avoided too. Achieve and demonstrate repeatedly, and IT operations becomes seen as less of cost-center and more as a value-generator, adding to the bottom line.

Continuous High-Quality Feedback

Enabling a continuous cycle of feedback is fundamental to an agile operations shift-left approach and the success of a DevOps program. In its simplest form, feedback from operations should help development toward reducing the volume of code defects, while feedback from development should guide operational service-level requirements before an application goes into production.

That's easy to say but difficult to achieve in practice. Applications are more diverse, distributed, and ephemeral, meaning feedback must be faster, richer, useable, and useful. To support this, modern approaches must clinically remove noise, distill intelligence, and then (and this is the really important part) put it in context of the people having the most to benefit from it. This way feedback is more valuable because it's actionable.

---

[5] "Omnichannel, Microservices, and Modern Apps," January 2016: http://www.ca.com/content/dam/ca/us/files/white-paper/ema-omnichannel-microservices-and-modern-applications.pdf

[6] Settling the Settling the Breadth vs. Depth: http://www.ca.com/content/dam/ca/us/files/ebook/settling-the-breadth-vs-depth-debate-mfinfrastructure.pdf

Take for example a mobile shopping app update with new social network API that's going to be released to take advantage of Black Friday. The new API is potentially a great way to increase revenue, but with demand difficult to predict, how do developers know that their code will handle back-end latency and load? We could take the "suck it and see" approach or stack the data center with more capacity, but with brand reputation on the line, is that a risk worth taking?

Alternatively, we could use agile operations thinking—serving up developers' critical insights into API latency, back-end load issues, end-to-end transaction times, mobile device performance impacts, every time they commit their code. This way, feedback is provided exactly when they have the most to gain from it—as they develop. And since it's placed in context of the business goals they're looking to support, it's immediately actionable. This enables the entire team to act with more purpose and urgency.

Scenarios like this play out all the time in the real world, with agile operations approaches providing the cross-functional glue needed to build a shared understanding of both problems and opportunities for improvement. For example, by using monitoring tools in performance benchmarking, teams can better address capacity and scalability issues; gaining confidence, new applications under development can cope with production load. This was a significant benefit that Danish Retailer Dansk Supermarked realized when implementing an APM solution to monitor development, test, and production environments.[7]

By facilitating the sharing of performance information, agile operations help teams move beyond making small incremental improvements in operational efficiency toward becoming a business differentiator. This is evidenced by a Techvalidate report, which indicated that more than half of organizations using APM stated it has helped them proactively manage user experience to create competitive advantage.[8]

Intelligence and Analytics

As operational functions become fused with development, advanced analytics and statistical methods will play a key role in serving DevOps practitioners the essential information needed to drive improvements.

Rather than attempting to process a mountain of alerts to gain clarity over the current state of performance, cross-functional teams will newer techniques to better predict application performance and usage. Armed with these insights, teams will make smarter and faster decisions, not just in a production context, but across the software lifecycle (see Figure 7-2).

---

[7] http://www.ca.com/content/dam/ca/us/files/case-studies/dansk-supermarked-group-safeguards-online-sales-with-ca-apm.pdf

[8] https://www.techvalidate.com/product-research/ca-application-performance-management

Some valuable methods include:

- *Mobile app analytics crash reports*—Help ensure any reliability issues impacting user experience are quickly resolved. With device and platform specific analytics, developers can optimize code for uniform performance across all mobile endpoints.
- *Behavioral analytics*—By understanding how mobile apps perform under real-world conditions (e.g., slow wireless networks), developers can optimize the user experience (UX) designs.
- *Usage and performance analytics*—These enable developers to improve design with every new iteration, ensuring what they build stays performant as usage and conditions change.
- *Business analytics*—These can increase the chances of an app meeting business targets when it moves into production. Analytics can also be used to measure ROI from different applications to help prioritize future development (e.g., revealing how newly developed features and functions are helping increase customer engagement and preventing churn).

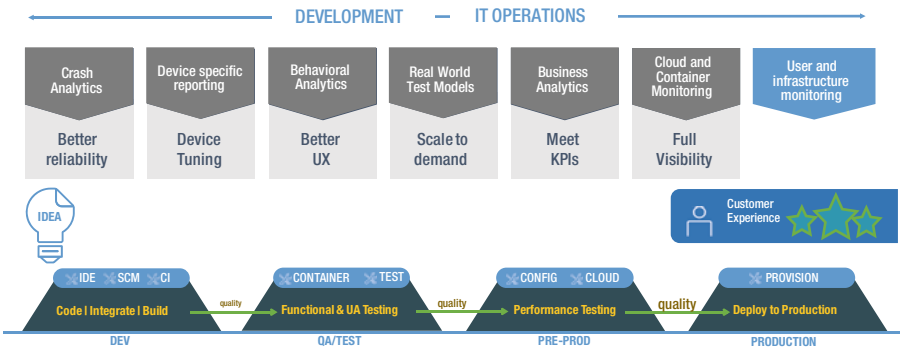

Figure 7-2. Agile operations analytics—quality and resilience across the software factory

With the digital-business landscape shifting continuously, predictive analytics will be supplemented with prescriptive techniques that can actually guide practitioners toward finding the best course of action in any given situation. And with less and less time to conduct protracted root-cause exercises, techniques like these will become essential for agile operations.

In an APM context, it's easy to see immediate usefulness for complex triage—prescribing solutions as problems emerge. But as these technologies mature, they'll also become more business-centric. Consider for example having the ability to *predict* a missed online sales target due to a slow developing performance problem and then quickly *prescribing* necessary changes in both application (design, function, etc.) and infrastructure to prevent the problem from occurring.

Providing business outcome-based capabilities like these will become the ultimate litmus test of successful agile operations and a shift-left strategy.

Agile Operations Tooling

To deliver agile operations and shift-left approaches we've described, organizations need to ensure they have the right application performance management strategies and tooling. Before looking at the tools themselves, and as illustrated in Figure 7-3, let's outline some major considerations:

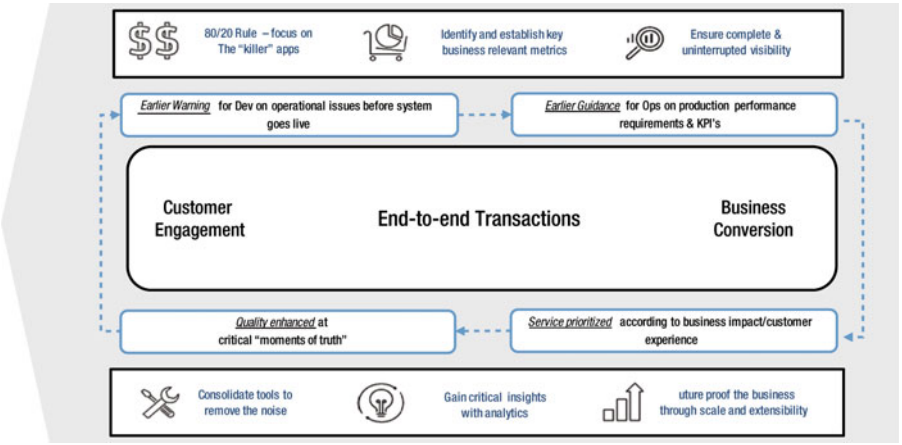

Figure 7-3. Agile operations and shift-left monitoring—essential strategies and tools capabilities

Early Warning for Business and Development

Agile operations tools should provide key insights into the performance of business transactions before the system goes live. With business traversing APIs, mobile apps, web infrastructure, and back-end systems-of-record, comprehensive views into performance across the entire application and infrastructure fabric become essential. This enables development teams to quickly understand infrastructure dependencies, how functional changes impact business performance, and where refactoring is needed.

By way of example, let's consider how integrating APM with continuous integration tools (e.g., Jenkins) can help developers. By providing APM data during a build process, developers can quickly determine the impact their code will make on performance (the early warning) and then leverage it to enact improvements. For example:

- Immediately see API usage increases between builds. *Payback:* Avoid increased costs in API pay-per-use scenarios.
- Identify authentication service overuse in new mobile app development. *Payback:* Improve customer experience.
- See how upstream functional changes cause downstream performance issues. *Payback:* Determine risks associated with the build.

The APM data provided should also be extremely granular (e.g., methods and transactions) and allow developers to roll back to see the impact of any code-level or environmental changes on performance. Additionally, by incorporating APM with continuous integration pass-fail build tests, high quality is established early (see Figure 7-4).

■ **Note** Using bi-directional integration, this process also updates APM, allowing operations to establish a complete performance "system of record" before production.

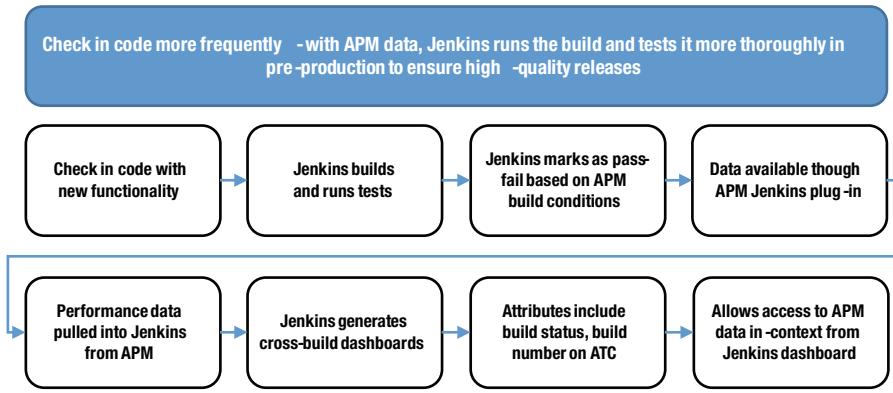

Figure 7-4. APM and Jenkins integrated workflow

In a mobile app context, agile operations tools should incorporate capabilities that help business and development teams better understand and respond to customer behaviors during mobile engagement. Here, end-to-end transaction

performance monitoring should be supplemented with analytics that reveal how new functions are driving increased engagement levels, activity, and customer retention—data that is critical when aligning development efforts to business initiatives.

Early Guidance on Operational Impact

This involves employing tools that continuously track business transactions and customer engagement to guide operations on production system requirements and KPIs. This requires discussion and shared agreement on what's important and why, collaboratively developing a shared set of metrics against which the system as a whole (comprising application and infrastructure components) can be monitored.

Difficulties may arise when architectural complexity and new emergent application behaviors make it difficult to establish accurate baselines against which application performance can be measured. Too often, best-guess performance baselining results in intermittent but acceptable performance spikes (caused by new application functionality) flooding monitoring systems with alerts. In other cases, longer-lived problems indicative of serious code defects go unnoticed because they fall within tolerance levels. To address this, teams should consider mechanisms that rely less on subjective or best-guess baselining, preferring instead statistical methods such as differential analysis where patterns of performance are built and monitored using multiple weighted criteria. Then, when thresholds are breached, deeper analysis (such as transaction tracing) can be initiated to pinpoint the problem and provide immediate and accurate feedback.

Prioritize by Business Impact and Customer Experience

Today's applications rarely function in isolation. In an online flight booking system, the overall customer experience can be delivered by 10 or more discrete elements delivered across multiple channels—from selecting a seat online and checking luggage at a kiosk, to making a payment and scanning a boarding pass with a mobile app.

The elements supporting this "experience" could be transactional, contextual, or a mixture of both. Applications and infrastructure will be equally rich, including APIs, mobile apps, wireless networks, sensors, and kiosks, along with legacy booking and CRM applications. It's essential therefore that agile operations tools deliver deep diagnostics across all these technologies, plus the ability to unify and monitor information at a higher service level. In the case of a flight booking system, for example, being able to prioritize a seemingly unimportant API latency problem because of its significance in supporting a new revenue generating service (e.g., a car rental booking service from a partner).

Feedback at Key Moments of Truth

In all environments, agile operations tools should provide teams fast feedback on software code effectiveness and problem components. This feedback is most critical at key points impacting customer experience (moments of truth). By leveraging live app experience analytics, for example, operations can provide invaluable information to developers and business analysts about the effectiveness of new code in improving user engagement, activity, and retention.

In cloud environments where application resources are invoked according to demand, cost, time, or other business metrics, feedback may be more difficult. Often, when performance problems arise the first question asked is "what changed," which can lead to finger pointing and conflict. While many tools can detect changes, they still require lengthy analysis to determine if the change caused a problem. More modern solutions address this by placing changes in context of application performance, allowing teams to roll back to a point in time to determine what changed and the knock-on effect on performance.

To support these strategies, modern monitoring solutions should deliver the following capabilities:

- *Noise removal*—Distill and simplify complex application topologies into role-based views based on elements such as application component, location, and business unit. Using these and other attributes, developers and support analysts can quickly reorient toward addressing issues and enacting improvements in context of the task at hand.
- *Analytical insights*—As discussed, provide all stakeholders actionable information across the software lifecycle. With app experience analytics, for example, advanced geo-spatial services allows business analysts to build performance and usage patterns, while developers can use video app playback to better determine the impact of newly introduced functionality on customer experience.
- *Uninterrupted visibility*—Deliver end-to-end transaction level visibility from the mobile app to the mainframe. This visibility should be provided in from a customer experience perspective with full and immediate traceability into problem areas across the supporting applications and infrastructure (e.g., network response time and API calls).
- *Scale and extensibility*—The only way to effectively engage customers and transact more business at scale is through mobile and cloud. Monitoring therefore needs to be equally scalable—future-proofing business by seamlessly supporting new technologies as they're introduced.

- *Manage to outcomes*—Strong consideration should be given to advanced tools that better determine what customer experience improvements (business outcomes) can be achieved by aggregating technology diagnostics and optimizing performance (outputs). This is key to ensuring the actions of cross-functional teams are fully aligned toward driving business improvement—not managing technology for technology's sake.

Summary

The days of IT operations working in the closed confines of a network operations center and just keeping the technology "lights" on are numbered.

As software releases grow in volume, variety, and velocity, agile operations will emerge as a key DevOps enabler. It's a collaborative discipline where practitioners work across the software factory to help business achieve the best outcomes from a high-quality customer experience.

Practitioners of agile operations will become true DevOps craftsmen; employing new methods, modern tools, and advanced analytics to deliver superior application performance without increasing the support burden.

In the next chapter, we'll discuss how DevOps can coexist with existing methodologies and best practices. We'll also examine its impact on other IT functions, including enterprise architecture and security.